แบบฝึกเรื่อง กองซ้อน

วันพฤหัสบดีที่ 11 สิงหาคม 2565

1. จงเขียนโปรแกรมในการดำเนินการกับโครงสร้างข้อมูลกองซ้อนตามข้อกำหนดดังนี้

   1.1 stack ใช้สำหรับเก็บข้อมูลจำนวนเต็ม เป็นตัวแปรชนิดแถวลำดับ

   1.2 เขียนฟังก์ชันย่อยในการ push และ pop กับ stack

   1.3 ให้โปรแกรมทำงานแบบวนรอบเพื่อรับข้อมูลตัวเลือกการทำงาน (x) จากผู้ใช้ โดยมีความหมาย ดังนี้

   - x = 1 ทำการรับค่าจำนวนจริง y แล้วเรียกใช้ฟังก์ชัน push โดยส่งค่า y เป็นพารามิเตอร์ พร้อมทั้งแสดงข้อความว่า Push ตัวเลขลงในสแตก
   - x = 2 เรียกใช้ฟังก์ชัน pop เพื่อนำข้อมูลที่ได้จากการ pop มาแสดงในหน้าจอ
   - x = 0 เรียกใช้ฟังก์ชัน pop เพื่อนำข้อมูลจากสแตกทั้งหมดมาแสดงผลในหน้าจอ และสิ้นสุดการทำงานของโปรแกรม
   - x เป็นจำนวนเต็มใด ๆ ที่ไม่ใช่ 0, 1, 2 ให้แสดงข้อความ “input error” แล้วแสดงเมนูและรับค่าอีกครั้ง

<u>ตัวอย่าง การรันโปรแกรม</u>

```
Enter choice 1. push 2. pop 0. exit: 1
Input real number: 2.3
PUSH 2.3 into a stack

Enter choice 1. push 2. pop 0. exit: 1
Input real number: 5
PUSH 5.0 into a stack

Enter choice 1. push 2. pop 0. exit: 2
POP 5.0 from a stack

Enter choice 1. push 2. pop 0. exit: 5
Input error

Enter choice 1. push 2. pop 0. exit: 1
Input real number: 88.1
PUSH 88.1 into a stack

Enter choice 1. push 2. pop 0. exit: 3
You want to exit !!!
POP 88.1, 2.3
BYE BYE….
```

ให้ส่งงานในระบบ TSU MOOC ภายในชั่วโมงปฏิบัติการ

2. จงเขียนโปรแกรมเพื่อทำการรับนิพจน์ในรูปของ postfix expression เพื่อนำมาคำนวณหาผลลัพธ์ของนิพจน์

โครงสร้างข้อมูล/ตัวแปร ที่ใช้

- stack เป็นสแตกสำหรับเก็บค่าจำนวนเต็ม (เป็นตัวแปรชนิดแถวลำดับ)
- a, b ตัวแปรที่มีชนิดข้อมูลเป็นจำนวนเต็ม
- op ตัวแปรชนิดอักขระ ใช้สำหรับอ่านอักขระจากนิพจน์

ขั้นตอนวิธีของการคำนวณนิพจน์ postfix

1) อ่านตัวอักขระจากนิพจน์ postfix มาทีละตัวอักษร เก็บไว้ใน op
2) ตรวจสอบตัวอักขระ op ที่อ่านเข้ามา
   2.1) ถ้าตัวอักขระที่อ่านมาเป็น '#' (หมายถึงสิ้นสุดการอ่านนิพจน์)
      2.1.1) ให้ pop ข้อมูลออกจาก stack เพื่อนำมาแสดงผล
      2.1.2) สิ้นสุดขั้นตอนวิธี
   2.2) ถ้าตัวอักขระที่อ่านมาเป็น ตัวดำเนินการ (ตัวเลข)
      2.2.1) ให้ push ตัวเลขนั้นลงใน stack
      2.2.2) ย้อนกลับไปทำข้อ 1)
   2.3) ถ้าตัวอักขระที่อ่านมาเป็น ตัวกระทำ (เครื่องหมาย)
      2.3.1) pop ข้อมูลจาก stack มาไว้ที่ตัวแปร b
      2.3.2) pop ข้อมูลจาก stack มาไว้ที่ตัวแปร a
      2.3.3) คำนวณผลลัพธ์จากการดำเนินการตามเครื่องหมาย op ในรูปแบบ a op b
      2.3.4) push ผลลัพธ์ที่ได้จากข้อ 2.3.3 ลงใน stack
      2.3.5) ย้อนกลับไปทำข้อ 1)

ข้อกำหนดของการอ่านนิพจน์

- ตัวถูกกระทำ (operand) เป็นตัวเลขโดด (0 – 9) ที่มีค่าบวกเท่านั้น
- ตัวกระทำ (operator) เป็นเครื่องหมายที่ใช้ดำเนินการกับตัวถูกกระทำ ประกอบด้วย สัญลักษณ์ '+' แทนการบวก '-' แทนการลบ '*' แทนการคูณ '/' แทนการหาร และ '^' แทนการยกกำลัง
- สัญลักษณ์ '#' แทนการสิ้นสุดนิพจน์
- นิพจน์ postfix จะต้องเป็นนิพจน์ที่ถูกต้องเสมอ

ตัวอย่าง Input/Output ของโปรแกรม

Input Expression:	Operation	Output:
23+45-*#	push(2), push(3) b = pop(), a = pop() c = a+b, push(c)	2 + 3 = 5 [c = 5]
23+45-*#	push(4), push(5) b = pop(), a = pop() c = a-b, push(c)	4 – 5 = -1 [c = -1]
23+45-*#	b = pop(), a = pop() c = a * b, push(c)	5 * (-1) = -5 [c = --5]
23+45-*#	c = pop()	-5
34*5-111+*^#	3,4,*,5,-,1,1,1,+,*,^,# 12,5,-,1,1,1,+,*,^,# 7,1,1,1,+,*,^,# 7,1,2,*,^,# 7,2,^,#	3 * 4 = 12 12 – 5 = 7 1 + 1 = 2 1 * 2 = 2 7 ^ 2 = 49
536*-96*7-+#	5,3,6,*,-,9,6,*,7,-,+,# 5,18,-,9,6,*,7,-,+,# -13,9,6,*,7,-,+,# -13,54,7,-,+,# -13,47,+,#	3 * 6 = 18 5 – 18 = -13 9 * 6 = 54 54 – 7 = 47 (-13) + 47 = 34

ข้อกำหนดของโปรแกรม

- การ push และ pop ให้เขียนในรูปของ ฟังก์ชัน
- ให้แสดงรูปแบบและผลลัพธ์ของการคำนวณทุกครั้ง
- ทดสอบการทำงานของโปรแกรมโดยใช้ตัวอย่างที่แสดงไว้ด้านบน